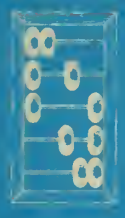
510.84
IL6r
no.1139-1145
1983-84
INC.
cop.3

510.84
I2L6
No. 1142

Report No. UIUCDCS-R-83-1142 UILU-ENG 83 1724

# Mutual Consistency Maintenance in A Prototype Data Traffic Management System

August 31, 1983



THE LIBRARY OF THE
FEB
UNIVERSITY OF ILLINOIS
CHAMPAIGN

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. UIUCDCS-R-83-1142

# Mutual Consistency Maintenance in A Prototype Data Traffic Management System

by

G. G. Belford, J. W. S. Liu, D. Cho, P. Cotten, S. England
J. D. Goldstein, S. C. Hwung, K. A. Kaufman, C. K. Kim
J. Leo, A. P. Manolas, A. Moon, A. D. Smet, Y. L. Yan

Department of Computer Science
University of Illinois
1304 W. Springfield Avenue
Urbana, Illinois 61801

and

G. L. Robinson and R. L. Lapp

Department of the Army
Construction Engineering Research Lab.
Interstate Research Park
Newmark Drive, P. O. Box 4005
Champaign, Illinois 61820

August 31, 1983

This work was partially supported by the U. S. Army Construction Engineering Research Lab., Champaign, Illinois, Contract No. DACW88-81-C-0011.

## TABLE OF CONTENTS


| | | |
|---|---|---|
| Abstract | | i |
| List of Figures | | ii |
| Chapter 1. | Introduction | 1 |
| Chapter 2. | Background and Mission | 3 |
| Chapter 3. | Prototype Data Traffic Management System | 6 |
| Chapter 4. | The Linkage Directory | 9 |
| Chapter 5. | Mutual Consistency Maintenance Subsystem | 13 |
| Chapter 6. | Summary and Conclusions | 25 |
| References | | 26 |
| Appendix A | List of Abbreviations | 27 |
| Appendix B | UCP and TUTHP in the AMPRS II-to-CAPCES Link | 28 |

## ABSTRACT

This report describes the mutual consistency maintenance mechanism used in a prototype Data Traffic Management System (DTMS) which was designed and implemented to integrate several independent and stand-alone data systems owned by the U. S. Army Corp of Engineers (CE). The major design constraints were that no modifications to the data systems involved were allowed and that an update to each duplicate copy was to be handled as automatically as possible and in a manner approved by the administrator of the data system containing the copy. The purpose of this work was to develop and test operating procedures which can be expanded and modified so as to be applicable in the implementation of an operational version of DTMS. The capabilities of the prototype DTMS and the Mutual Consistency Maintenance Subsystem (MCMS) are discussed. The linkage directory used to support mutual consistency maintenance and the mutual consistency maintenance mechanism are described.

## LIST OF FIGURES


| Figure No. | | Page No. |
|---|---|---|
| 1 | Configuration of the Prototype DTMS | 7 |
| 2 | Hierarchical Structure of the Linkage Directory | 10 |
| 3 | General Structure of the MCMS | 14 |
| 4 | Project Data Input from AMPRS II | 17 |
| 5 | Description File | 18 |
| 6 | Examples of terminal displays and user responses | 21 |
| 7 | Acknowledgement message to AMPRS II user | 23 |

# CHAPTER 1. INTRODUCTION.

In many distributed systems, duplicate copies of files or records are kept at separate sites for reasons ranging from improvement in reliability and query response time to preservation of independent ownership of information gathered at different sites. Many update schemes have been developed [1 - 6] to ensure mutual consistency. It is typical to assume that the individual sites are connected via dedicated lines in the absence of network and host failures and that updates to all sites can be carried out automatically under the control of a distributed database management system or a distributed operating system.

This report describes the mutual consistency maintenance mechanism used in a prototype Data Traffic Management System (DTMS), which was designed and implemented to integrate several independent and stand-alone data systems owned by the U. S. Army Corps of Engineers (CE). Some of these data systems reside on a single host computer, but others are on remote hosts. Communication facilities provided to support host-to-host communication primarily consist of dial-up lines.

While the problem of maintaining consistency among several data systems on different host computers presents no serious difficulties in theory, the ideal solutions to the problem may not be implementable in a real situation because of political constraints. In the case of this pilot project, different data systems are operated and maintained by different organizations. Because authorization and access rights reside with the database administrators and users of the individual data systems, it is not always possible to update duplicate copies automatically. In fact, the constraints imposed on the design of the prototype DTMS were that no modifications to the data systems served by DTMS are allowed and that updates to duplicate copies must be handled in ways approved by the administrators of the individual data systems. Because of these constraints, the solutions described in [1 -6] were not applicable.

The prototype DTMS was designed and implemented to demonstrate the feasibility of integrating independent data systems to form a distributed data system in which mutual consistency is maintained between the component data systems and coherent access to more than one data system is supported. Specifically, the purpose of this project was to develop a general framework within which the mutual consistency maintenance mechanism and distributed query processing facility can be tailored to meet

the constraints of the individual data systems. Our primary objective was to develop and test operating procedures which can be expanded and modified so as to be applicable in the implementation of an operational version of a DTMS serving a dozen or more major data systems.

To provide the necessary background, Chapter 2 describes the motivation and mission of the project. Chapter 3 discusses the capabilities of the prototype DTMS, in general, and of the Mutual Consistency Maintenance Subsystem (MCMS), specifically. Chapter 4 describes the structure of the linkage directory that was implemented to support mutual consistency maintenance. Chapter 5 describes the mutual consistency maintenance mechanism used in the prototype MCMS. Our conclusions are summarized in Chapter 6.

# CHAPTER 2. BACKGROUND AND MISSION

This chapter is an expansion of the objectives and scope stated in the previous chapter. Within this chapter, the work on DTMS is placed within the context of the mission of the CE and its responsible offices in *project management*, i.e., the management of design, construction, operation, maintenance, and disposal of military facilities throughout the facility life cycle. The terminology used in this report is also defined here.

## 2-1. Project Data

In order to carry out its tasks in project management, the CE requires information on all associated facilities from facility inception, through construction completion (or real estate acquisition), and through disposal for Army-owned facilities. The information on the facilities is also used by other organizations and agencies to support a wide range of activities during the facility life cycle. Currently, the information on different projects is stored in a large number of stand-alone data systems. Data items relating to a single project stored in one data system are referred to collectively as the *project data* for that project.

Each data system has an *administrator*. In this case, the administrator is an organization element responsible for the operation of the data system and the maintenance of the data base stored in the system. The administrator controls data base access, issues user identification, and manages password procedures. He has full accountability for the adequacy of the data system and the accuracy of the information stored in the system. The *users* of the data systems are project managers. Users are granted write access to the project data for their projects by the administrator.

## 2-2. Redundancy in Stored Project Data

Since different organizations at different levels of project management collect and maintain the data stored in different data systems, redundancy in stored data is unavoidable. Moreover, due to different design objectives, the data systems often support different data models, user views, and user interfaces. For example, different file formats and identifiers may be used for data on the same project. The same data item may also

be stored in different data systems in different formats.

During any phase of the life cycle of a facility, all updates of a data item that is part of the project data related to this facility are first carried out in the data system used to support project management during that phase. Therefore, this copy of the data item may be considered as the *primary copy*. The user or the administrator who has the write access right to the primary copy of the data item is referred to as the *owner* of the data item. It is assumed that, at any one time, there is only one primary copy of any data item. Other copies of the data item stored in other data systems are *duplicate copies*. The user with the write access right to a duplicate copy of a data item is referred to as the *responsible user* of the data system containing that copy.

An update of a data item is carried out when the owner of the data item issues an *update (transaction)* in the data system containing the primary copy. This data system is referred to as the *sending data system* for that transaction. The data systems containing copies of one or more data items affected by an update transaction are referred to as the *receiving data systems*. An update transaction sent to a receiving data system to request update of a duplicate copy is referred to as a *triggered update (transaction)*.

### 2-3. Data Traffic Management System

Currently, updates to all duplicate copies of an item for the purpose of mutual consistency maintenance are applied manually upon receipt of information detailing the update that has been applied to the primary copy. This process is time-consuming and liable to error. Foreseeing the need to provide a more effective mutual consistency maintenance mechanism as well as a coherent interface to support report generation based on information residing in more than one data system, a Data Traffic Management System (DTMS) was planned. The functions to be supported by the DTMS can be categorized into the following two classes:

(1) Maintenance of mutual consistency between data systems ----- DTMS monitors updates to data systems served by it and determines whether data items updated have duplicate copies. When a data item with duplicate copies is updated, DTMS will submit an appropriately formulated triggered update transaction to each of the receiving data systems. The administrator of a receiving data system will be notified whenever triggered updates have been carried out if his system allows

automatically triggered updates. Alternatively, DTMS will inform him that the triggered update is needed and prompt him for appropriate actions. When a triggered update is carried out, an acknowledgement message will be sent to the owner of the data item. When a triggered update is deferred or rejected, the acknowledgement message will contain reasons for deferral or rejection.

(2) Report preparation and management decision support requiring access to information stored in more than one data system ----- The DTMS will provide support to facilitate queries to multiple data systems and generation of standard reports. A dictionary/directory system will be an integral part of the DTMS.

# CHAPTER 3. PROTOTYPE DATA TRAFFIC MANAGEMENT SYSTEM

This chapter provides an overview of the prototype DTMS. Its capabilities and configuration are described.

The prototype DTMS was designed to demonstrate the feasibility of (1) maintenance of mutual consistency between data bases stored in the data systems served by DTMS and (2) report generation and query processing requiring access to information stored in more than one data system. Two major design constraints were that no modifications to the data systems involved were allowed and that updates to duplicate copies triggered by an update of the primary copy were to be handled as automatically as possible, with minimal user intervention. The primary objective of this pilot project was that the procedures and mechanisms developed and tested in the prototype should be readily usable in an operational version of the DTMS.

The implementation of the prototype DTMS is partially completed. Its configuration is as shown in Figure 1. The portion shown in solid lines is implemented. The portions shown in dotted lines are still in the design stage. The DTMS currently resides on an IBM 370/3033 host computer operated by Tymshare, Inc. This computer is referred to hereafter as the *DTMS host*. The detailed characteristics of the individual data systems are not relevant. Three of these systems, the DD Form 1391 Processor, CAPCES, and AMPRS II, reside on the DTMS host and are supported by the database management system FOCUS. DEACONS runs on a Wang Labs VS90 computer which operates in IBM 370 emulation mode. HQ-IFS runs on an IBM compatible computer under the NOMAD data management system. The AR415-17 runs on a Honeywell 66/80 computer. The COEMIS F&A systems reside on Honeywell 66/20 computers.

The (Mutual Consistency Maintenance Subsystem) MCMS is implemented to enforce mutual consistency between AMPRS II, CAPCES, and DEACONS, as shown by the solid boxes in Figure 1. MCMS is being expanded to include the DD Form 1391 Processor to the list of data systems served. These data systems were chosen to be served by the prototype MCMS in order to demonstrate that the mechanisms used in the prototype MCMS can enforce mutual consistency between data systems both on and not on the DTMS host. A small and efficient directory is kept by the DTMS to support maintenance of mutual consistency. This directory, referred to as the *linkage directory* hereafter, contains information on locations of and access paths to duplicate copies of

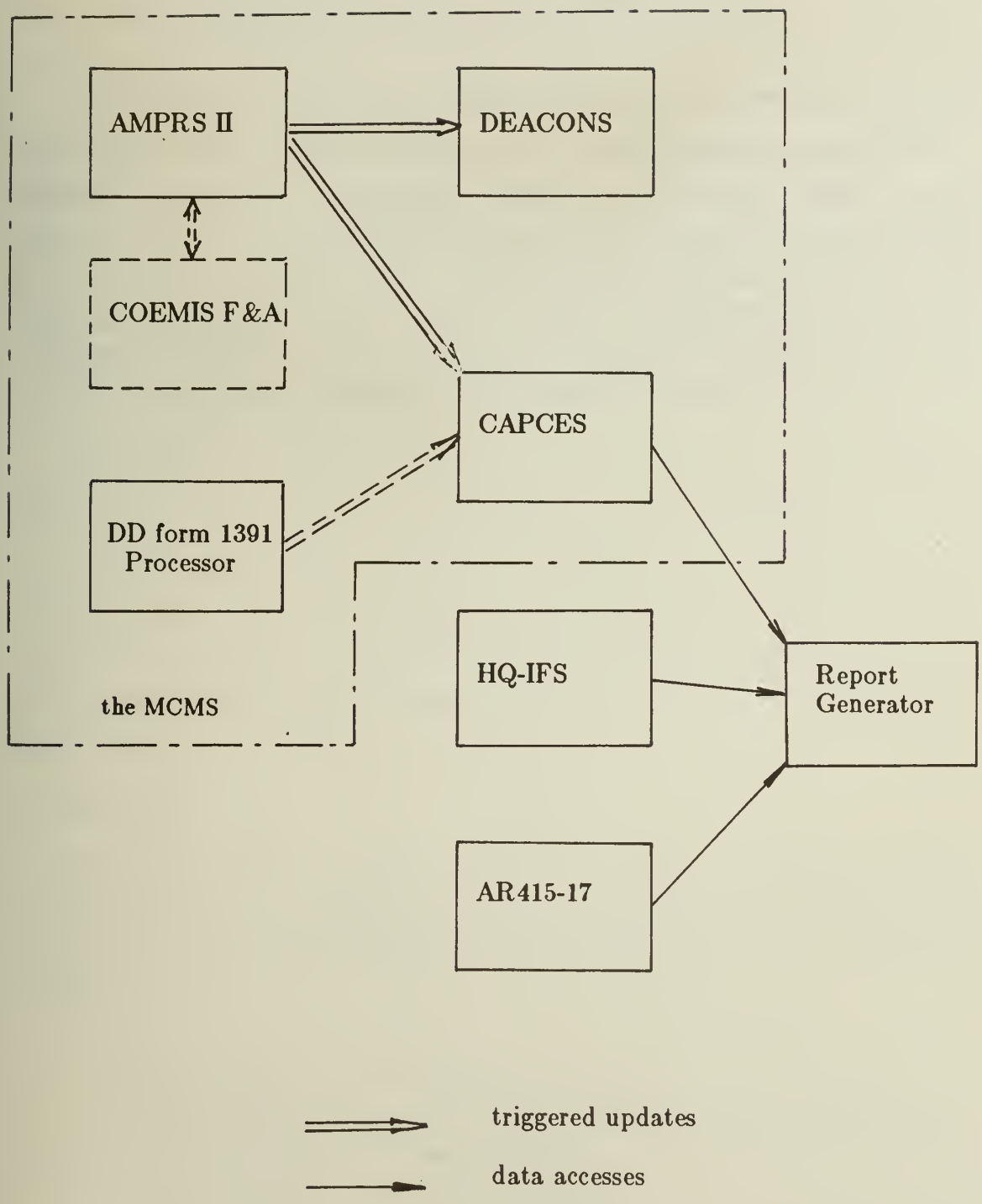



Figure 1. Configuration of Prototype DTMS

all data items. The structure of the linkage directory is discussed in Chapter 4. The general mutual consistency maintenance mechanism used in MCMS is described in chapter 5. The examples implemented in the pilot project are also described there.

The report generation portion of the prototype DTMS is in the design stage. Various design alternatives are being considered. Commercially available dictionary/directory systems are being evaluated to determine their suitability to be the basis of the DTMS dictionary/directory system. Facilities to support file extraction and transfer from remote systems are being investigated. As an experiment, a procedure that processes extract files containing cost data from AMPRS II, CAPCES, and HQ-IFS was implemented. Detailed information on this experiment can be found in [7].

# CHAPTER 4. THE LINKAGE DIRECTORY

In the prototype DTMS, the Data Traffic Management Directory (DTMD) consists of two subdirectories. The *linkage directory*, containing information on redundantly stored data items, is used to support maintenance of mutual consistency. The *dictionary/directory* contains all pertinent facts about all of the data items in the data systems served by the DTMS. It is used to support standard and ad hoc report generation and query processing that requires access to data stored in more than one data system. This chapter describes the structure and design of the linkage directory. The characteristics of the dictionary/directory system are discussed in [7].

The purpose of the linkage directory is to provide information on where duplicate copies of data items are stored and information on how triggered updates to each copy of the data items are handled. A primary design objective of this directory is that it allows searches of its contents to be done fast and efficiently. Hence, a minimal amount of data required to support mutual consistency maintenance is stored here. For example, metadata on any data item are kept in the linkage directory only when it is known to DTMS that there are two or more copies of the data item stored in the data systems served by DTMS.

## 4-1. Structure.

The linkage directory consists of 4 segments with a total of ten fields. Its hierarchical structure is as shown in Figure 2. The field names are tabulated below. The first field of each segment is a key field which can be used as a key in searching the directory.

| | |
|---|---|
| DB_NAME | data base (system) name |
| ITEM_NAME | name used in data system to identify the data item |
| ITEM_FORMAT | data item format |
| ITEM_DESCRI | data item description |
| UPDATE_PROC | update procedure name |
| USER_ID | name of user responsible for the data item |
| LINK_DB | name of receiving data system |
| LINK_ITEM | data item name in receiving data system |
| MISC_NO | |

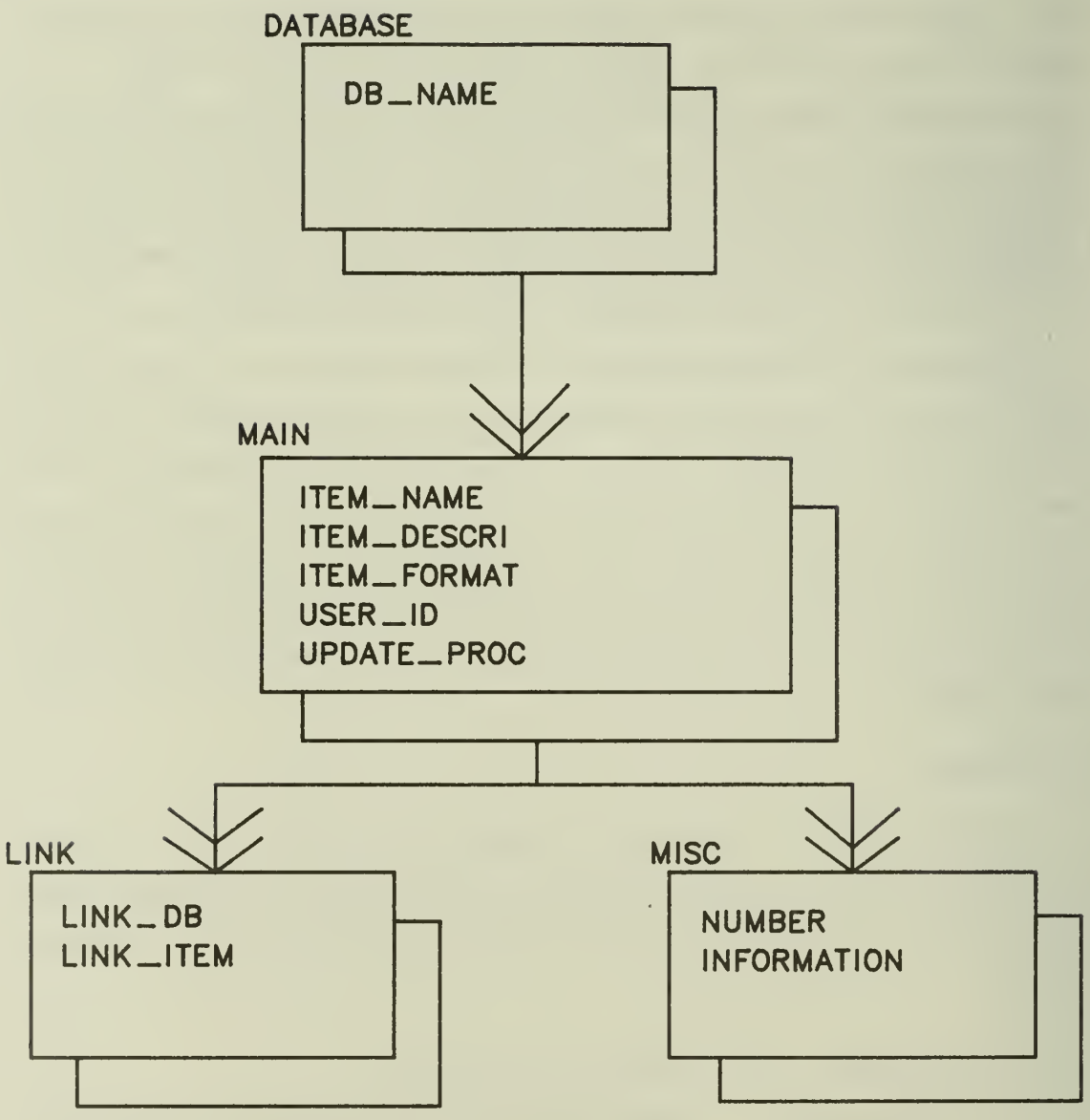



Figure 2. Hierarchical Structure of the Linkage Directory

MISC special comments

The key field of the DATABASE segment is DB_NAME (database name). For every data system served by the DTMS, there is an occurrence of a DATABASE segment with its name in the key field. For each duplicated data item in this data system, there is one occurrence of the child segment, MAIN. The MAIN segment is keyed by the ITEM_NAME field, which contains the name used to identify the data item in the data system. The values in the other fields, ITEM_DESCRI, ITEM_FORMAT, USER_ID and UPDATE_PROC, provide information on the format in which the data item is stored, the user who is responsible for updating the item, and the name of a DTMS procedure which updates this item in the data system. These metadata are used to aid in the formulation of triggered update transactions for this data item. The names of the data systems in which the data item is duplicated and its names in these data systems are specified by the metadata in the child LINK segments. The fields in the MISC segments are reserved fields for later expansion. In the prototype linkage directory, the MISC field is sometimes used to keep special comments. For example, when a data item has two duplicate copies in the same receiving data system, this fact is noted here.

**4-2. Access and Maintenance Procedures.**

A procedure GETLINK has been developed to retrieve information from the linkage directory. The program listings are on-line on the DTMS host and can be found in [8]. GETLINK is called by the update capture programs. For each data item updated in the sending data system, the update capture program for that sending data system calls this procedure to determine whether this data item is duplicated elsewhere and to retrieve the names of the receiving data systems and the corresponding names of the data item in these data systems. The calling parameters include the sending data system name and data item name. This procedure checks the linkage directory for the data system specified by the data system name and upon finding it, proceeds to search for the given data item. If either the data system or the data item is not found, an appropriate code is returned to the calling program. When both the data system and the data item are found in the directory, a link segment is returned to the calling program. This procedure is called repeatedly until there are no more links for the given data item and an appropriate code indicating this fact is returned to the calling program. At this time, all receiving data systems are identified.

Since the linkage directory is central to mutual consistency enforcement, it is important to keep the directory accurate and up-to-date. For this purpose, a menu-driven procedure for additions to and modification of the linkage directory and a batch procedure for bulk loading of the directory have been implemented. Using the menu-driven linkage directory maintenance procedure to modify the directory interactively, the user is prompted to enter all the necessary information in order to ensure the consistency of the directory [9]. For more efficient bulk loading, the user may use the bulk loading procedure by first creating an input file of updates and then simply typing the command:

UPBULK <input file name> <input file type>

The procedure UPBULK carries out a number of integrity checks, and any errors discovered are logged into an error listing for subsequent correction.

# CHAPTER 5 MUTUAL CONSISTENCY MAINTENANCE SUBSYSTEM

This chapter describes the MCMS in the prototype DTMS, which is designed and implemented to maintain mutual consistency between AMPRS II and CAPCES and between AMPRS II and DEACONS. These two components of the subsystem are referred to as the AMPRS II-to-CAPCES link and AMPRS II-to-DEACONS link, respectively. In this subsystem, the sending data system is AMPRS II. The receiving data system is either CAPCES or DEACONS.

## 5-1. General Structure of MCMS

The general configuration of the MCMS is as shown in Figure 3. For each of the data systems served by DTMS, there is an update capture process (UCP) and a triggered update transaction handling procedure (TUTHP). Mutual consistency is maintained in the following manner: The UCP of each data system determines whether any updates originated in the system require that triggered updates be carried out in other data systems. When a triggered update is required, the TUTHP of the receiving data system is called to carry out the update in the manner approved by the administrator of that system. Triggered updates may be carried out automatically or semiautomatically by DTMS, or manually with the help of DTMS. The two methods implemented in the prototype MCMS are described later on in this section.

### Update Capture

Depending on the manner in which the sending data system is interfaced to DTMS, the UCP may extract information on updates from update transaction files or from history files maintained by the sending data system. The UCP may also extract update information by monitoring the interactive work sessions during which update transactions are entered. The UCP may be waked up to run by the DTMS monitor at regular time intervals. Alternatively, it may be waked by a notification signal from the sending data system at the time when an update is carried out in that system. Clearly, depending on the characteristics of the data systems involved and their interfaces with DTMS, many different types of UCP need to be provided.

When the UCP of a sending data system runs, it determines whether any new updates have been made since the last time it ran. For each new update, the UCP captures all relevant information on the update. By consulting the linkage directory, the

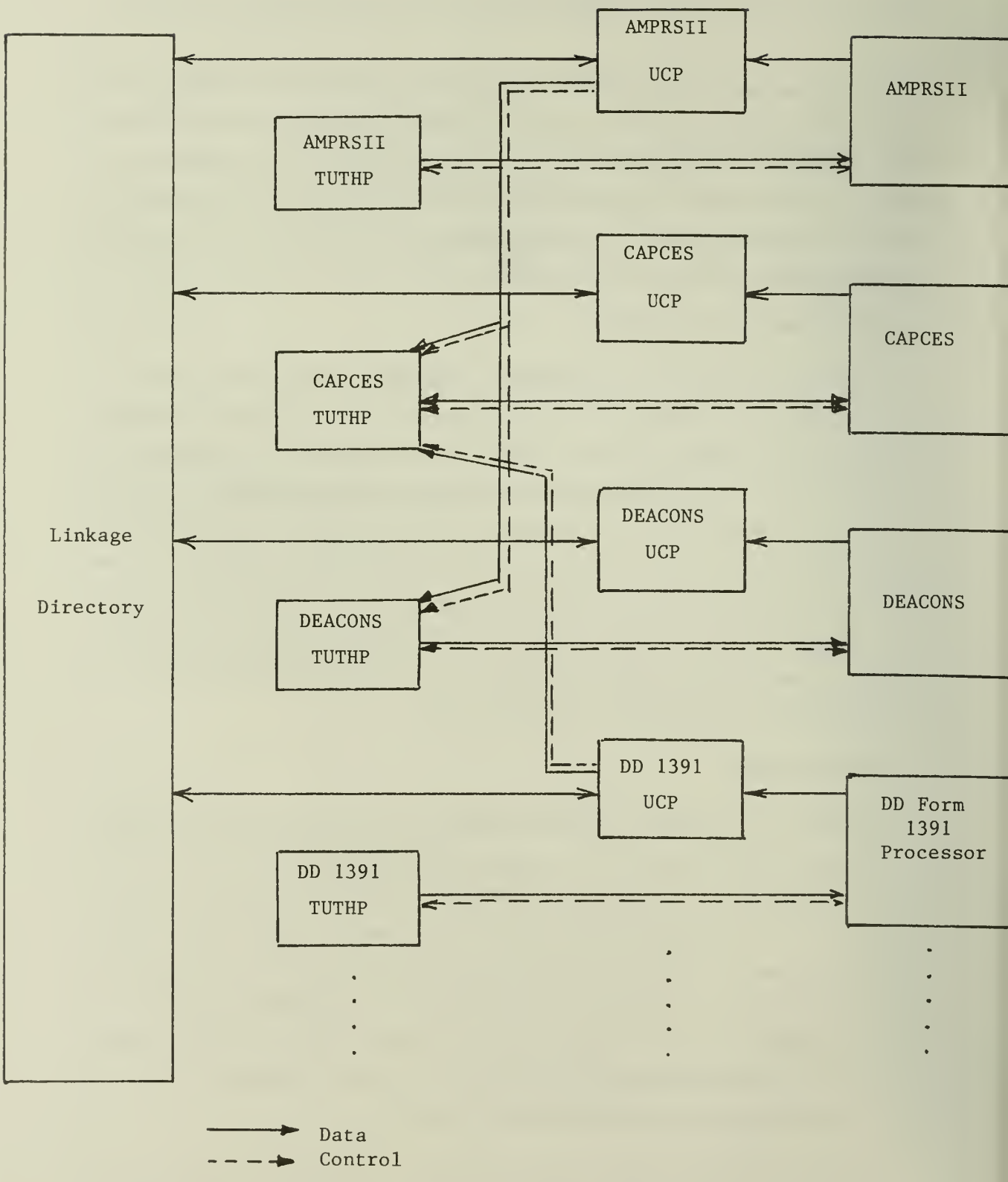



Figure 3. Mutual Consistency Maintenance Subsystem

UCP determines whether there are duplicate copies of the data item stored in other data systems. If there are, the UCP finds from the linkage directory the item name and format and the responsible user id in each of the receiving data systems. Necessary conversions in formats and values are carried out also.

The result generated by the UCP for each update transaction is an *input file* for each receiving data system. Each input file contains complete information needed by the receiving data system to carry out the triggered updates of all data items whose primary copies were updated by the update transaction. Because triggered updates are handled in different ways, the contents and formats of the input files are different for different receiving data systems. In the case where DTMS is allowed to carry out the triggered updates automatically or semiautomatically after permission is granted by the user, the input file is used directly as the input to the TUTHP of the receiving data system. In the prototype MCMS, automatic triggered updates were not allowed. In this case, a corresponding *description file* is also generated by the UCP. The description file is used as a part of the message sent to the responsible user of the receiving data system giving him the details on the triggered update transaction and requesting his permission to carry it out. The description file is also used as a part of the acknowledgement message sent by DTMS to the originating user.

**Triggered Update Transaction Handling**

When the TUTHP of a receiving system is called, each input file (and, if applicable, the corresponding description file) is processed in turn. For each of the data items contained in the input file, the TUTHP retrieves the needed access path information from the linkage directory. Based on this information, an appropriately formulated update transaction is generated and submitted to the receiving data system. The transaction may be submitted automatically by DTMS. Alternatively, as in the case of the AMPRS II-to-CAPCES link (described later), information such as the new and old values of each data item is presented to the responsible user. The user may choose to have the triggered update carried out, deferred, or rejected. If he chooses to carry out the triggered update, the triggered update is formulated and submitted for him. On the other hand, if he chooses not to carry out the triggered update, he is prompted to type in reasons for his action. In either case, the acknowledgement sent to the originating user contains information on his action and reasons.

In the case where the triggered update transactions to a receiving data system are handled completely automatically by MCMS, the TUTHP of that system is called by the UCP of the sending data system. In the case where triggered updates are handled in batch mode, the TUTHP is periodically awakened. When a triggered update transaction is submitted to and carried out by the receiving data system, the duplicate copy in that system is updated. Clearly, triggered updates should not in turn trigger more updates. Hence, a fully automatic system would require some method, such as a flag in the linkage directory to indicate which is the primary copy, to prevent cyclic update triggering. In the examples described below, triggered updates are handled semiautomatically under user supervision, since the administrators of the receiving data systems will not allow triggered updates be carried out automatically.

## 5-2. The Mechanism Implemented in the Prototype MCMS

The prototype MCMS is designed and implemented to maintain mutual consistency between AMPRS II and CAPCES and between AMPRS II and DEACONS. The AMPRS II UCP and the CAPCES TUTHP are described in detail in Appendix B.

### Interface with AMPRS II.

When an AMPRS II user initiates an update, the information required to carry out the update is placed in an update transaction file. This file is processed when the update to the AMPRS II data system is actually carried out. This may be either immediately after the transaction is submitted or after it has been batched together with other update transactions. The AMPRS II UCP interfaces with AMPRS II via the AMPRS II update transaction files. At regular intervals, the AMPRS II UCP copies the update transaction files from the AMPRS II user space into the DTMS space. By processing the update transaction file, the UCP determines the data items updated and their new values.

### Triggered Update Handling in the AMPRS II-to-CAPCES Link.

When the receiving data system is CAPCES, the information extracted from a update transaction file is as listed in Figure 4. Since the CAPCES database administrator does not allow completely automatic triggered updates, triggered updates are handled in the following semiautomatic manner: The corresponding description file, shown in Figure 5, is used as a part of a message which is displayed for the CAPCES

Sending Data System Identification
User (Project Manager, Sender) Identification
Project Identification (Including Items Below, as Applicable)
    Station (Location) Code (Standard)
    Project Number Symbol
    Type Funds Symbol
    Authorization (Program) Year (Last Two Digits)
Project Data Item Name-Value Pairs
    Name (or Number) 1, Value 1
    Name (or Number) 2, Value 2
    .  .
    .  .
    Name (or Number) n, Value n

Figure 4. Data extracted from AMPRS II

user who is responsible for updating the data item in CAPCES. It is also inserted in the acknowledgement message returned to the AMPRS II user who initiated the update.

Due to the limitation of 80 characters per line on most CRT terminals, each of the data values in a description file is at most 15 characters in length. Similarly, due to the facts that CRT terminals on Tymshare systems operate in scroll mode and that there are only 24 lines per display screen, it is not desirable to display any file containing more than 24 lines. (It will be difficult for a user to read the lines in the beginning of the file as the lines scroll upwards if there are more than 24 lines to be displayed.) For this reason, the number of data items in each description file is limited to five in the prototype AMPRS II-to-CAPCES link. It is noted that this limitation need not be imposed in the operational version of the DTMS. A more sophisticated window manager which presents consecutive segments of the displayed file to users as individual pages or allows scrolling one line at a time under user control may be developed for the operational version of the DTMS. Such a window manager will provide a more suitable user interface for displaying files of arbitrary lengths.

For every data item in the description file displayed to him, the responsible CAPCES user is given the choice of having the copy of the data item in CAPCES updated as in AMPRS II, of refusing to carry out the update, or of deferring the update until some later time. If his choice is to update the data item, the update to CAPCES is automatically carried out for him. Otherwise, he is prompted to type in reasons for refusal or deferral. In the following, the interaction between the CAPCES user and the DTMS are described.

(1). Informing the CAPCES User. The CAPCES user is notified of an incoming message from DTMS immediately upon sign-on. A message with the following format is displayed on his terminal when he signs on.

DTMS MESSAGES ARE WAITING.
DO YOU WANT MESSAGES DISPLAYED?

(2). Display of Description File and Prompt for Action. The user responds by typing in yes (or y) or no (or n).

(i). If "no" (or "n") is the response, the short message (described in (1)) and the description file(s) are stored for the user. The short message is displayed each time he

| **Receiving Data System** | **Sending Data System** |
|---|---|
| CAPCES | AMPRS II |
| CAPCES User ID | AMPRS II User ID |
| Project ID in CAPCES | Project ID in AMPRS II |
| Station (Location) Name | Station (Location) Code |
| Project Number Symbol | Project Number Symbol |
| Type Funds Name | Type Funds Code |
| Authorization (Program) Year | Authorization (Program) Year |
| Project Description (Title) | Category Code (F4C) |

Data items to be updated in CAPCES

| No. | Name in CAPCES | New Value | Original Value | Name in AMPRS II | Value in AMPRS II |
|---|---|---|---|---|---|
| 1 | Name (No.) 1 | Value 1 | Value 1 | Name (No.) 1 | Value 1 |
| 2 | Name (No.) 2 | Value 2 | Value 2 | Name (No.) 2 | Value 2 |
| - | - | - | - | - | - |
| - | - | - | - | - | |
| 5 | Name (No.) 5 | Value 5 | Value 2 | Name (No.) 5 | Value 5 |

Figure 5 Format of Description File

signs on.

(ii). If the user types "yes", the terminal displays the following:

DATE YY MM DD TIME hh:mm:ss
MSG NO. DTMS Message Identification Number

The description file corresponding to this msg. no. (See Figure 5)

SELECT ONE OF THE FOLLOWING ACTIONS:

1. ACCEPT ALL ITEMS
2. DEFER ALL ITEMS
3. REJECT ALL ITEMS
4. ACCEPT IN PART

(3). Further Interactions. (i) If the selection of the user is either 1, 2, or 3, the terminal displays 1, 2, or 3, respectively, as shown below:

1. PROJECT DATA UPDATING COMPLETED
2. TYPE DEFERRAL REASON(S)
3. TYPE REJECTION REASON(S)

In the case of 2 or 3, the user types in reasons for rejection or deferral of the update of the data items in CAPCES. The following are examples of user responses to messages 2 and 3, respectively.

2. UPDATES TEMPORARILY DEFERRED TO STABILIZE OUR REPORTS
3. PROJECT WAS DELETED PER DIRECTIVE (NUMBER)

(ii). In the case where the user response is 4, descriptions of the data items are displayed one at a time. For each data item displayed, the user is prompted to type in his choice of action. Examples of the terminal displays and user responses are shown in Figure 6. Lines in upper case letters are displayed by the system and lines in lower case letters are user responses.

| LINE NO. | NAME IN CAPCES | NEW VALUE | ORIGINAL VALUE | NAME IN AMPRS II | VALUE IN AMPRS II |
|---|---|---|---|---|---|
| 1 | XY2XY2 | 42 | 41 | XY2 | 72 |

>ENTER 1 TO ACCEPT, 2 TO DEFER, OR 3 TO REJECT: 3
[TYPE REJECTION REASON(S): The project was deleted.
Blank line

| LINE NO. | NAME IN CAPCES | NEW VALUE | ORIGINAL VALUE | NAME IN AMPRS II | VALUE IN AMPRS II |
|---|---|---|---|---|---|
| 2 | PAD482 | 4003 | 2032 | PAD | 6000 |

>ENTER 1 TO ACCEPT, 2 TO DEFER, OR 3 TO REJECT: 2
[TYPE DEFERRAL REASON(S): update temporarily deferred to stabilize our reports on this project. update will be carried out before September 10, 1983.
Blank line

Figure 6. Examples of terminal displays and user responses

(4). Completion of All Triggered Updates. For each triggered update transaction to be carried out in CAPCES, there is a message formatted as described in (2.ii) waiting to be read by the CAPCES user. For each message, steps (2) and (3) are repeated until the following message appears.

NO MORE MESSAGES

Or, in the case where the user wishes to stop before all triggered updates are taken care of, unprocessed messages are stored for him to handle in the next session.

(5). Acknowledgement to AMPRS II User. For each triggered update transaction acted upon by the CAPCES user, an acknowledgment message is sent to the AMPRS II user identified in the user name field of the description file. This acknowledgement message informs the AMPRS II user which action has been taken by the responsible CAPCES user. Figure 7 summarizes the acknowledgement messages that might be received by the AMPRS II user.

The last line in Figure 7 is displayed only when the message in 4 indicates that updates to some data items are rejected. Whenever the response to the question DO YOU NEED ASSISTANCE is "yes", the user is prompted to type in the name and telephone number of the person to be contacted. Appropriate messages are then sent to the DTMS management office so that help may be provided to the CAPCES and AMPRS II users.

After a triggered update is acted upon by the responsible CAPCES user, the content of the description file for that triggered update is kept in one or more history files. More specifically, one history file contains information on updates in which some or all items were accepted. There are also two "wait" files, one of which contains descriptions of updates which were entirely or in part deferred or rejected and the other contains information in machine-readable form on deferred transactions, so that they can be processed later.

**Triggered Update Handling in the AMPRS II-to-DEACONS Link.**

DEACONS is the data system containing information on Air Force design and construction projects. Since the administrator of DEACONS does not permit DTMS software to be resident on the DEACONS host computer, the AMPRS II-to-DEACONS triggered updates must be applied to DEACONS in batch mode. Specifically, the

| ACTION | DISPLAY |
|---|---|
| | DATE YY MMM DD TIME hh:mm:ss<br>MSG NO. DTMS Message Identification Number<br>Corresponding description file |
| 1. | YOUR MESSAGE (NUMBER) IS ACCEPTED |
| 2. | YOUR MESSAGE (NUMBER) IS DEFERRED<br>REASON(S): UPDATES TEMPORARILY DEFERRED TO STABILIZE OUR REPORTS |
| 3. | YOUR MESSAGE (NUMBER) IS REJECTED<br>REASON(S): PROJECT WAS DELETED PER DIRECTIVE (NUMBER)<br>DO YOU NEED DTMO ASSISTANCE? |
| 4. | YOUR MESSAGE (NUMBER) IS TREATED AS FOLLOWS<br>LINE I DEFERRED (REASON: DATA ITEM VALUE TEMPORARILY ....)<br>LINE J DEFERRED (REASON: UPDATE TEMPORARILY DEFERRED ....)<br>LINE K REJECTED (REASON: SHOULD NOT DATE 980331 BE 890331?)<br>ALL OTHERS ARE ACCEPTED<br>DO YOU NEED ASSISTANCE? |

Figure 7. Acknowledgement message to AMPRS II user

DEACONS administrator wants simply a complete project description created from AMPRS II data and placed in a file accessible to him whenever any of the data items pertaining to an Air Force design and construction project are changed in AMPRS II. Hence, there are no DTMS interactions with the DEACONS user or direct applications of the updates as in the AMPRS II-to-CAPCES link. A "triggered update transaction" in the case of the AMPRS II-to-DEACONS link is therefore handled in the following manner: When the AMPRS II UCP finds that the project id of a transaction contained in an AMPRS II update transaction file is that of an Air Force project, the transaction file is written to the input file generated for the DEACONS TUTHP. In this case, the input file is in the same format as that of the AMPRS II update transaction file.

The DEACONS TUTHP is called to process the Air Force projects transactions file as follows. The linkage dictionary is checked to see if a transaction updates an item that requires a triggered update to DEACONS. For all such transactions, the project identifier is written to a file for further processing. This file is sorted by project identifier and duplicates are deleted. For each project identifier, a set of data element values are retrieved from AMPRS II. These values are converted to the formats of the corresponding data elements in DEACONS. They are then combined with the project identifier and written to an output file in a suitable format for automated processing by DEACONS. This output file is then spooled to the DEACONS user ID (on the DTMS host), from which it may be retrieved by the DEACONS user and applied to the DEACONS database. The details on the design and implementation of the AMPRS II-to-DEACONS link can be found in [10].

# CHAPTER 6 SUMMARY AND CONCLUSIONS

This report describes the MCMS of the prototype DTMS designed and implemented by the Department of Computer Science, University of Illinois at Urbana-Champaign, for the Army Construction Engineering Research Laboratory. The work on the prototype DTMS was carried out to demonstrate the feasibility of creating a DTMS which would serve as a control point to coordinate and integrate the major data systems owned and accessed by the Corps of Engineers. Once the feasibility of these procedures has been demonstrated, the prototype implementation can be expanded and modified so as to be applicable to the dozen or so major data systems to be served by DTMS.

Theoretically, the problem of maintaining consistency among copies of data items distributed among several data systems presents no major problems. Many solutions have been proposed. The lesson learned in this project was that ideal solutions to a problem may not be implementable in a real situation, not because of technical difficulties but because of political constraints. The design of the prototype DTMS was constrained by the requirements that no modifications to the data systems served by DTMS be made and that different triggered update handling mechanisms, as demanded by the administrators of of the data systems, be provided.

The design and implementation of the facilities needed to carry out several examples of DTMS processing were completed under this project. These facilities provide a general framework within which the mutual consistency maintenance mechanism can be tailored to meet the demands of the individual data systems. Specifically, the mutual consistency maintenance mechanism was investigated both in the case when both data systems reside on the DTMS host and when the sending data system is on the DTMS host but the receiving data system is remote. In particular, major tasks accomplished include the implementation of the linkage directory to support mutual consistency maintenance and a semiautomatic triggered update handling mechanism. The linkage directory, along with the procedures for its search and maintenance, can be easily expanded to support mutual consistency maintenance in the operational DTMS. The mechanism used to handle triggered updates in the prototype MCMS can be easily generalized to be used in the operational DTMS as well.

## REFERENCES

[1] Alsberg, P. A., G. G. Belford, J. D. Day, and E. Grapa, "Multi-copy Resiliency Techniques", in *Distributed Database Management,* edited by P. Bernstein, J. B. Rothie, and D. W. Shipman; IEEE, 1978.

[2] Ellis, C. A. "A robust algorithm for updating duplicate databases," *Proc. of 2nd Berkeley Workshop on Distributed Data Management and Computer Networks,* 1977.

[3] Kung, H. T. and J. R. Robinson, "On optimistic methods for concurrent control," *ACM TODS,* vol. 6, June, 1981.

[4] Lampson, B. and H. Sturgis, "Crash recovery in a distributed data storage." Tech. Report, Xerox PARC, 1976.

[5] Parker, D. S., et.al, "Detection of Mutual Inconsistency in Distributed Systems," *IEEE Trans. on Software Engineering,* Vol.SE-9, May, 1983.

[6] Thomas, R. F., "A solution to the concurrency control problem for multiple copy databases," *Proc. Spring COMPCON, Feb, 1978.*

[7] "Prototype Data Traffic Management System, " Final Report by Department of Computer Science, University of Illinois, August 31, 1983.

[8] Cotton, P., "The Implementation of a Pilot Data Traffic Manager System," MCS Project Report, Depart. of Computer Science, University of Ill., Urbana, Ill. 1981.

[9] Moon, A., "A directory to support automatic update triggering," MCS Project Report, Department of Computer Science, University of Ill., Urbana, Ill., 1983.

[10] A. N. Manolas, "AMPRSII - DEACONS data transfer", MCS Project Report, Department of Computer Science, University of Ill., Urbana, Ill., 1983.

# APPENDIX A
## LIST OF ABBREVIATIONS

| | |
|---|---|
| AMPRSII | Automated Military (Design and Construction Project Management) Progress Reporting System |
| AR415-15 | Military Construction, Army (MCA) Program Development |
| AR415-17 | (Empirical) Cost Estimating for Military Programming |
| CAPCES | Construction Appropriations Programming, Control, and Execution System |
| CE | U. S. Army Corps of Engineers |
| CERL | U.S. Army Construction Engineering Research Laboratory |
| COEMIS | Corps of Engineers Management Information System |
| COEMIS F&A | COEMIS, Finance and Accounting Subsystem |
| DD | Department of Defense |
| DEACONS | USAF Design and Construction System |
| DTMS | Data Traffic Management System |
| F&A | Finance and Accounting |
| FOCUS | A DBMS marketed by Information Builders, Inc. (IBI), resident on Tymshare, Inc.'s TYMCOM-370 |
| HQ | Headquarters |
| HQ-IFS | HQ Integrated Facilities System |
| IBM | International Business Machines, Inc. |
| MCMS | Mutual Consistency Maintenance Subsystem |
| PAX | Programming, Administration, and Execution System |
| TUTHP | Triggered Update Transaction Handling Process |
| UCP | Update Capture Process |

# APPENDIX B

## FLOW CHARTS AND DESCRIPTIONS OF UCP AND TUTHP IN AMPRS II-TO-CAPCES LINK

This appendix contains flow charts which describe the AMPRS II UCP and CAPCES TUTHP. The programs described here are also documented on-line on the DTMS host computer.

### B-1. AMPRS II UCP, DEMO3.

The input to the AMPRS II update capture process, *DEMO3*, are the AMPRS II update transaction files. Each of these files contains the following information: (1) Identification of the AMPRS II user who originated the update transaction, (2) Project identification consisting of station code, project number, type funds code, and authorization year, and (3) Project data item name-value pair for each of the data items updated. The output generated by DEMO3 are placed in the DESCFILE file, the content of which is later processed by *RCV EXEC*, the CAPCES TUTHP. The format of the DESCFILE file is shown in Figure B-1. For each AMPRS II update transaction, there is a segment of the DESCFILE file containing information in both the input file and the description file described in Chapter 5.

The operations DEMO3 are as shown in Figure B-2. Subroutines are marked by " * ". The control structure of DEMO3 is shown in Figure B-2a. At a regular interval, the DEMO3 picks up the notifications from AMPRS II users (each of which contains the initiator's user id and file name), and copies the update transaction files in the AMPRS II user space into the DTMS space. The DTMS file containing the AMPRS II update transaction files is named TI. For each item contained in this file, the linkage directory is consulted to determine whether the item is listed in the directory. The data item is listed in the linkage directory when it is redundantly stored in other data systems. In this case, for each receiving data system, DEMO3 finds from the directory the item name and format and the responsible user id. This step, Step 1, is illustrated by the flow chart in Figure B-2b. The output generated by this step is the TEMPFILE. The flow chart for the subroutine PROCESS-TRANS called in Step 1 is in Figure B-2c. The subroutine PROCESS-ITEM, which is called by PROCESS-TRANS, is described in Figure B-2d. DSEARCH, the subroutine which is equivalent to GETLINK, is called by

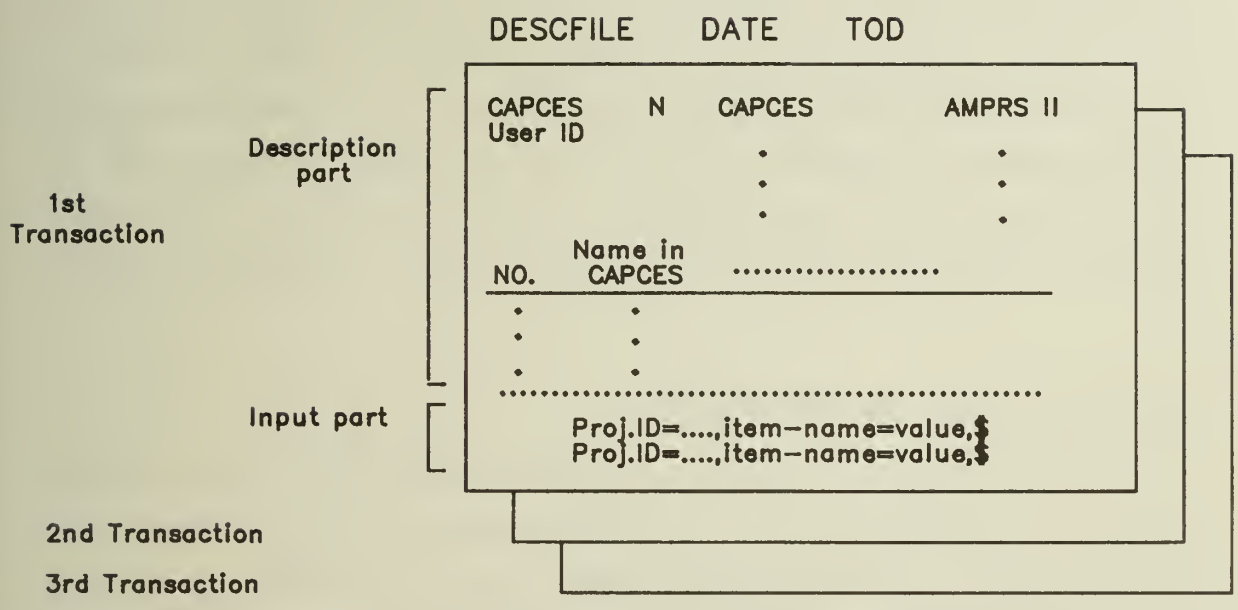



Figure B-1. DESCFILE File Format

PROCESS-ITEM to search the linkage directory. The flow charts of DSEARCH is in Figure B-2e.

Other necessary conversions such as project id , format of value, etc. are also carried out. The results obtained by processing the AMPRS II update transaction files are placed in TEMPFILE. TEMPFILE is then sorted by USER ID and PROJ ID. This sorted output is used as input to build the DESCFILE file. This step, Step 2, of DEMO3 is illustrated by the flow charts in Figure B-2f.

As Step 3, the DEMO3 checks for the existence of the old description files. If an old DESCFILE file exists, the DEMO3 checks the ACK file which contains the acknowledgement messages from the receiving users stating that their portions of DESCFILE file are already processed. Thus, segments of the DESCFILE that are already processed are determined. DEMO3 clears the processed segments of the old DESCFILE file, merges the the remaining portion of the DESCFILE with the newly created one, and adds time stamp to the DESCFILE. Step 3 is illustrated in Figure B-2g.

### B-2. Operations of the RCV EXEC.

Triggered updates to CAPCES are handled semiautomatically. The CAPCES TUTHP, RCV, is awakened to run when the responsible CAPCES user signs on and wishes to process the pending triggered updates. Figure B-3 shows the operations of RCV EXEC, which processes the DESCFILE file created by the update capture process, generates and sends the messages, and carries out updates to CAPCES as described in Chapter 5. SHOWCAP, CAPDATE, COMPLETE, and JFTNMS are called by RCV. The operations of RCV are described below:

(1) When the responsible CAPCES user enters RCV EXEC, a link is created to the DTMS space. The DESCFILE line pointer is set to the first line of DESCFILE STORE.

(2) If the DESCFILE line pointer has reached the end of file, the procedure -DONE is called (beginning at step 15.)

(3) The first item in the line pointed to in DESCFILE STORE is checked. If it is not the CAPCES user's user id, it means that the user has already processed this

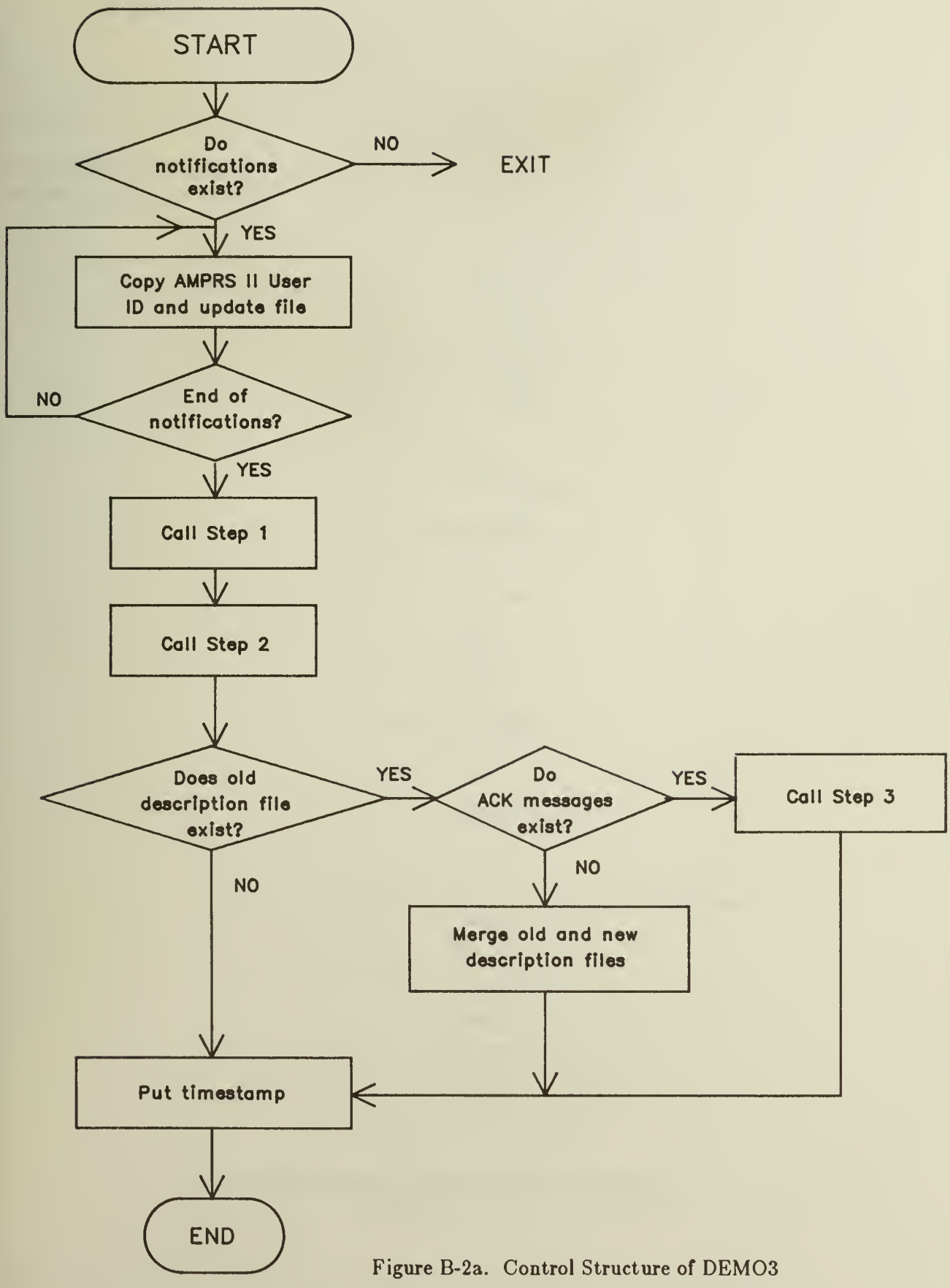



Figure B-2a. Control Structure of DEMO3

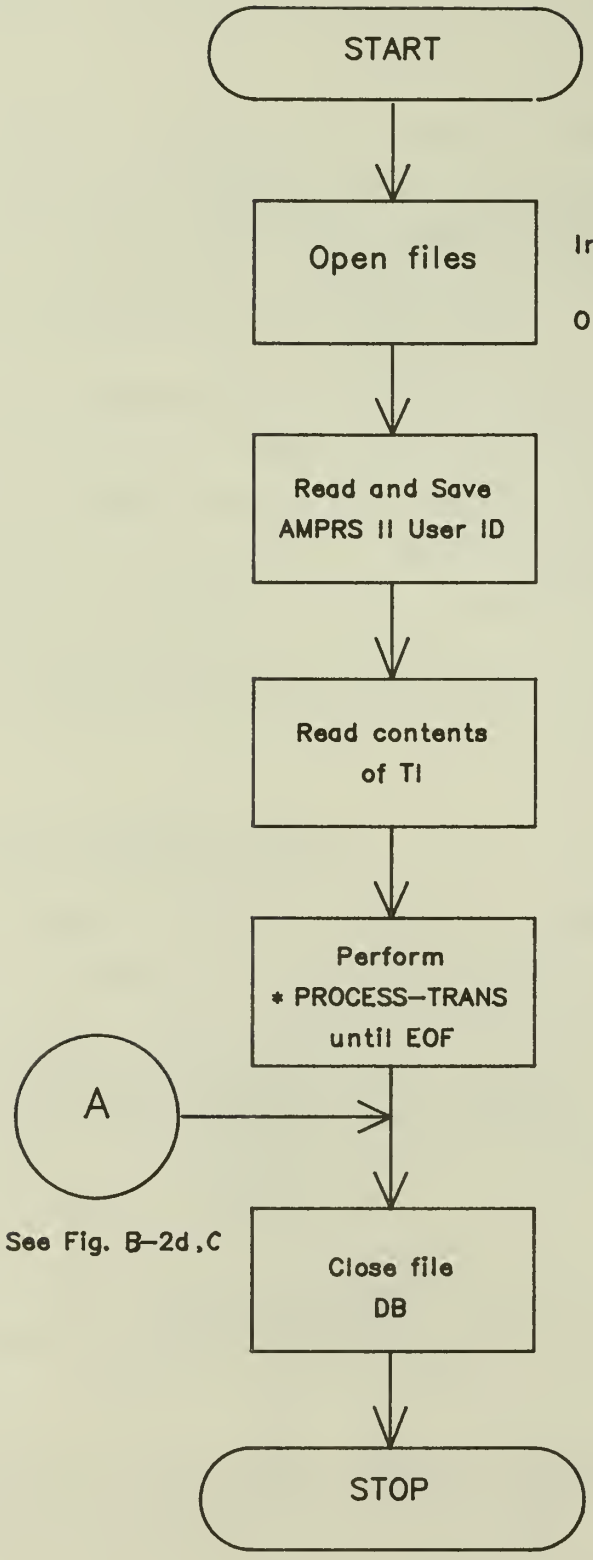



Figure B-2b. Flow Chart for Step 1 of DEMO3

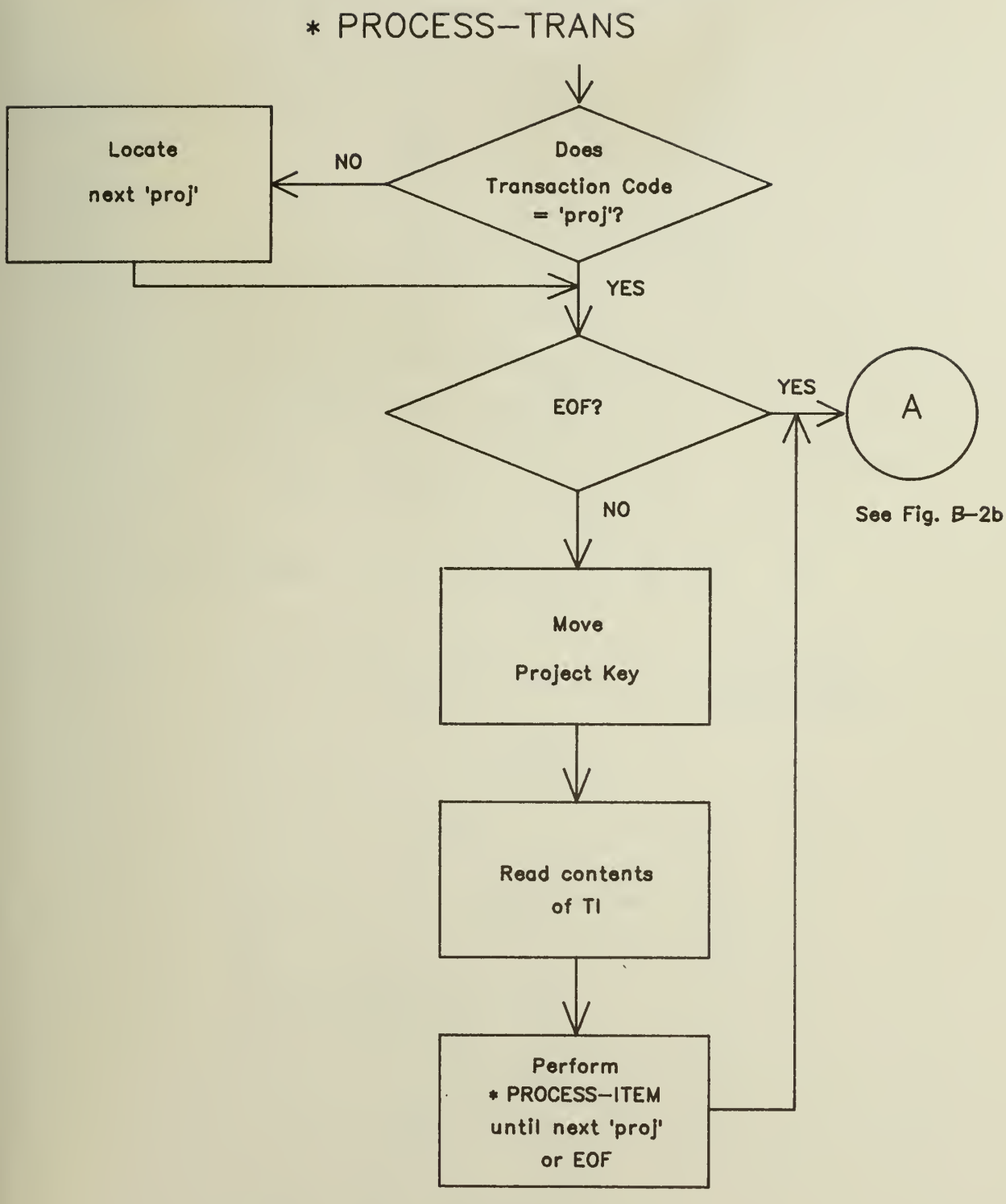



Figure B-2c. Flow Chart for PROCESS-TRANS
(called in Step 1)

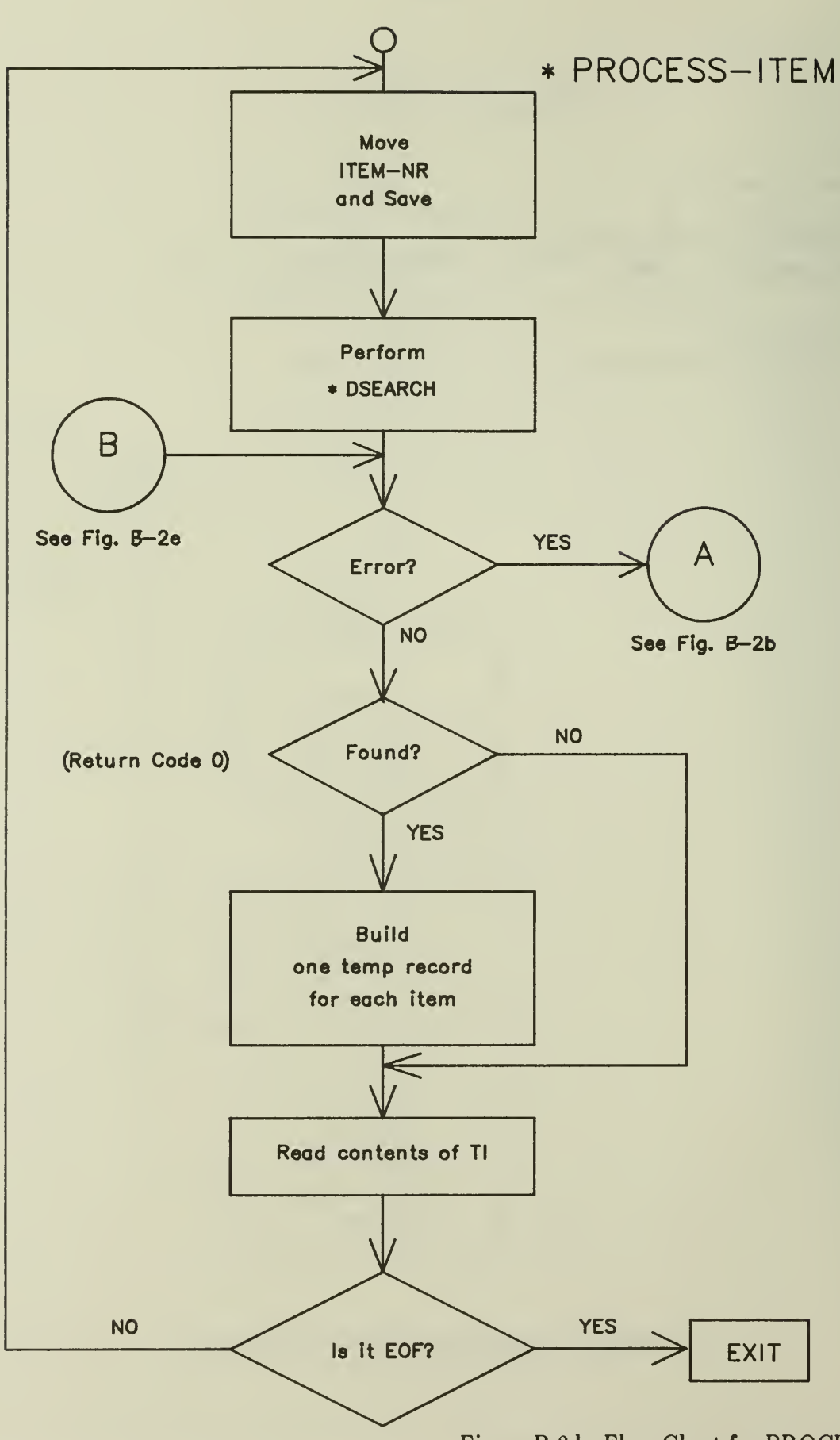



Figure B-2d. Flow Chart for PROCESS-ITEM
(called by PROCESS-TRANS)

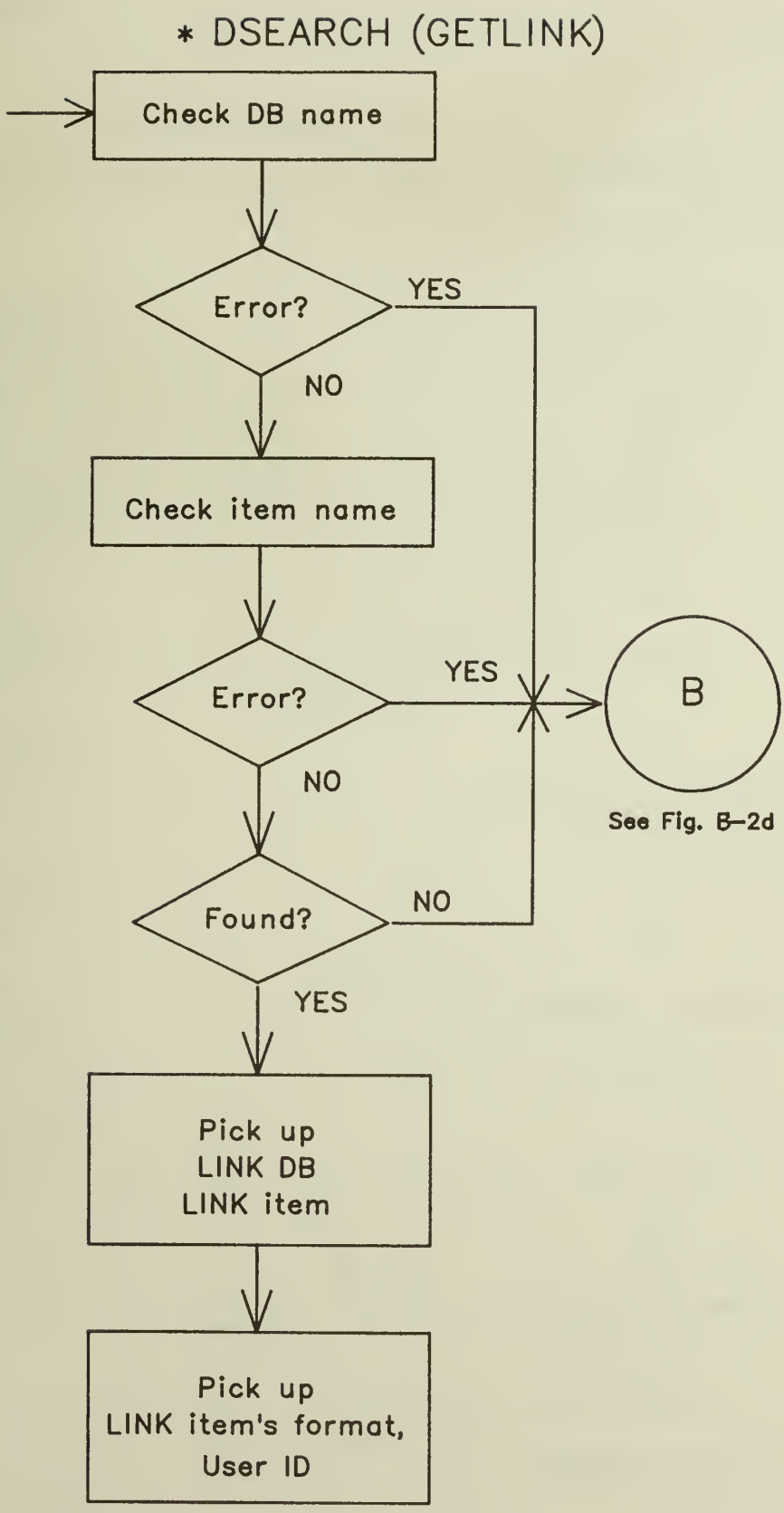



Figure B-2e. Flow Chart for DSEARCH (called by PROCESS-ITEM)

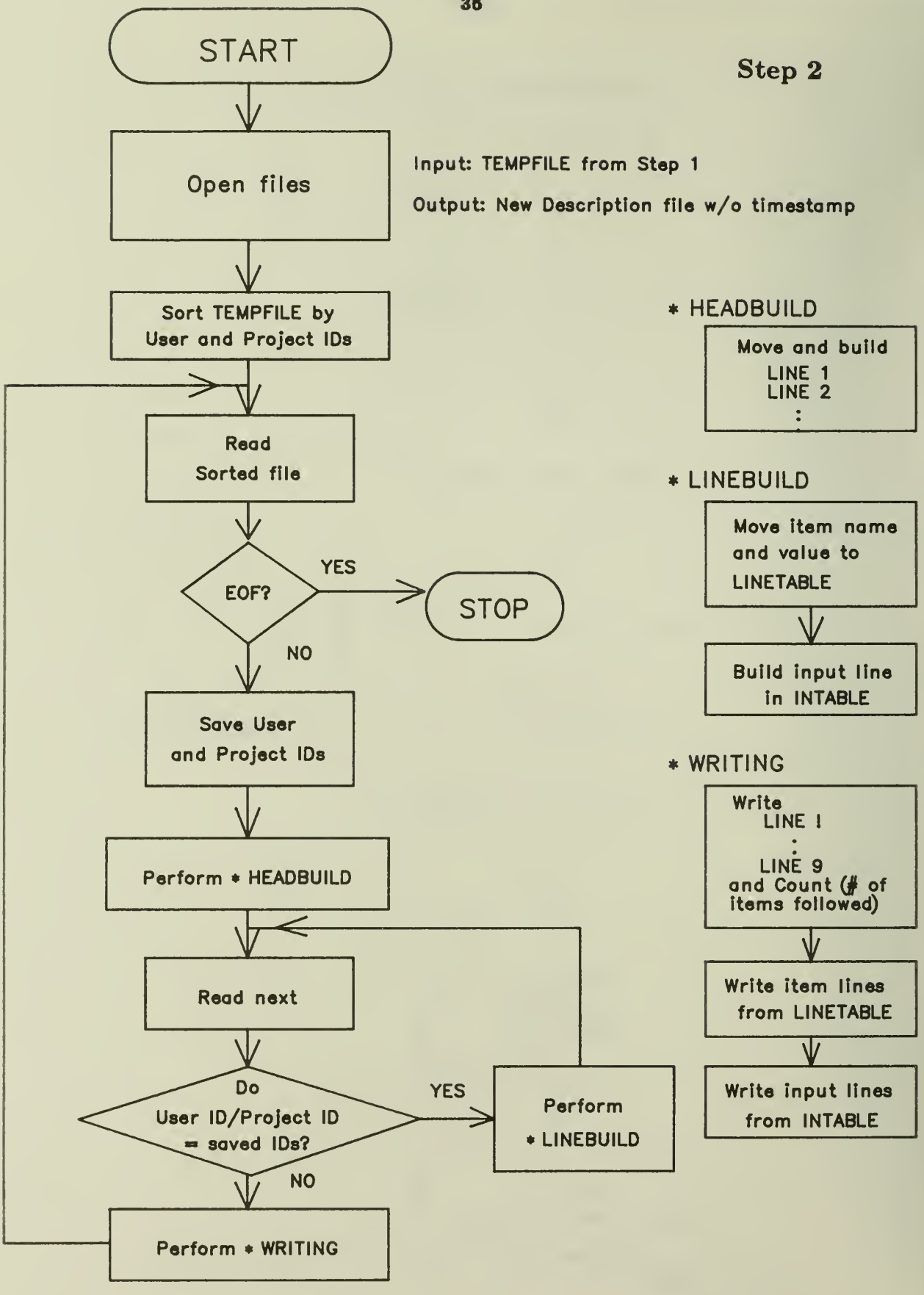



Figure B-2f. Flow Chart for Step 2 of DEMO3

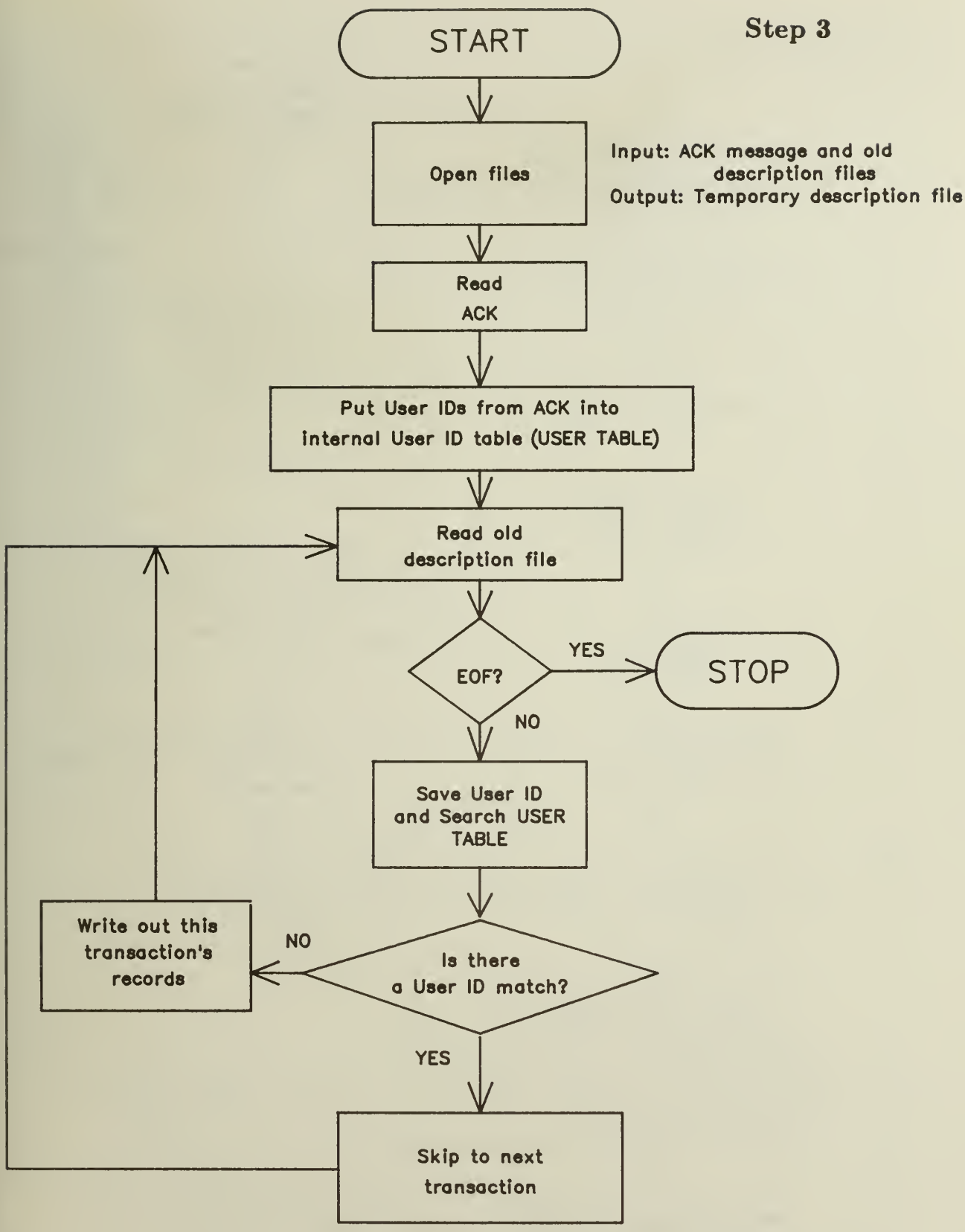



Figure B-2g. Flow Chart for Step 3 of DEMO3

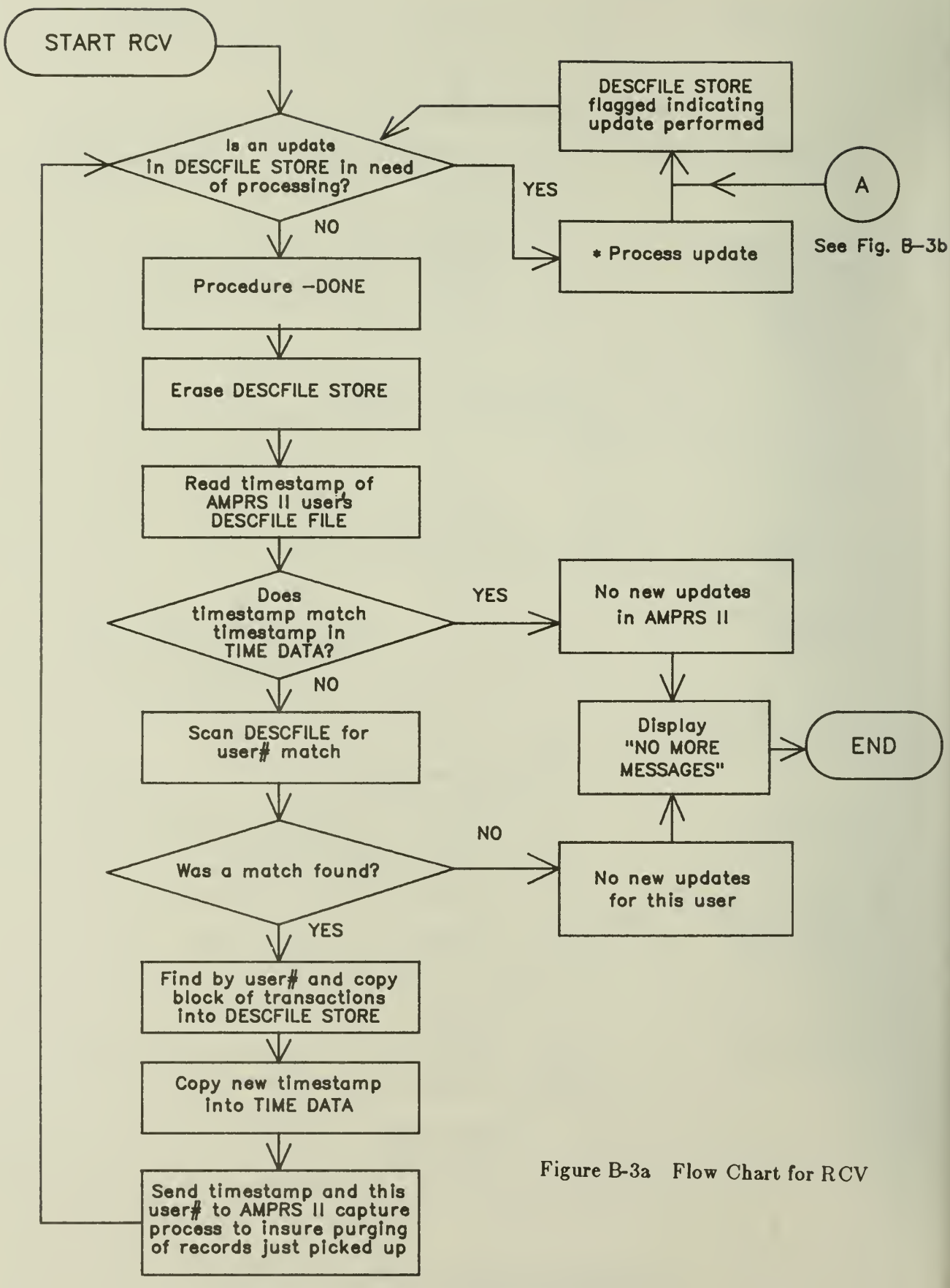

START RCV
Is an update in DESCFILE STORE in need of processing?
YES
NO
DESCFILE STORE flagged indicating update performed
A
See Fig. B-3b
* Process update
Procedure -DONE
Erase DESCFILE STORE
Read timestamp of AMPRS II user's DESCFILE FILE
Does timestamp match timestamp in TIME DATA?
YES
NO
No new updates in AMPRS II
Display "NO MORE MESSAGES"
END
Scan DESCFILE for user# match
Was a match found?
NO
YES
No new updates for this user
Find by user# and copy block of transactions into DESCFILE STORE
Copy new timestamp into TIME DATA
Send timestamp and this user# to AMPRS II capture process to insure purging of records just picked up

Figure B-3a Flow Chart for RCV

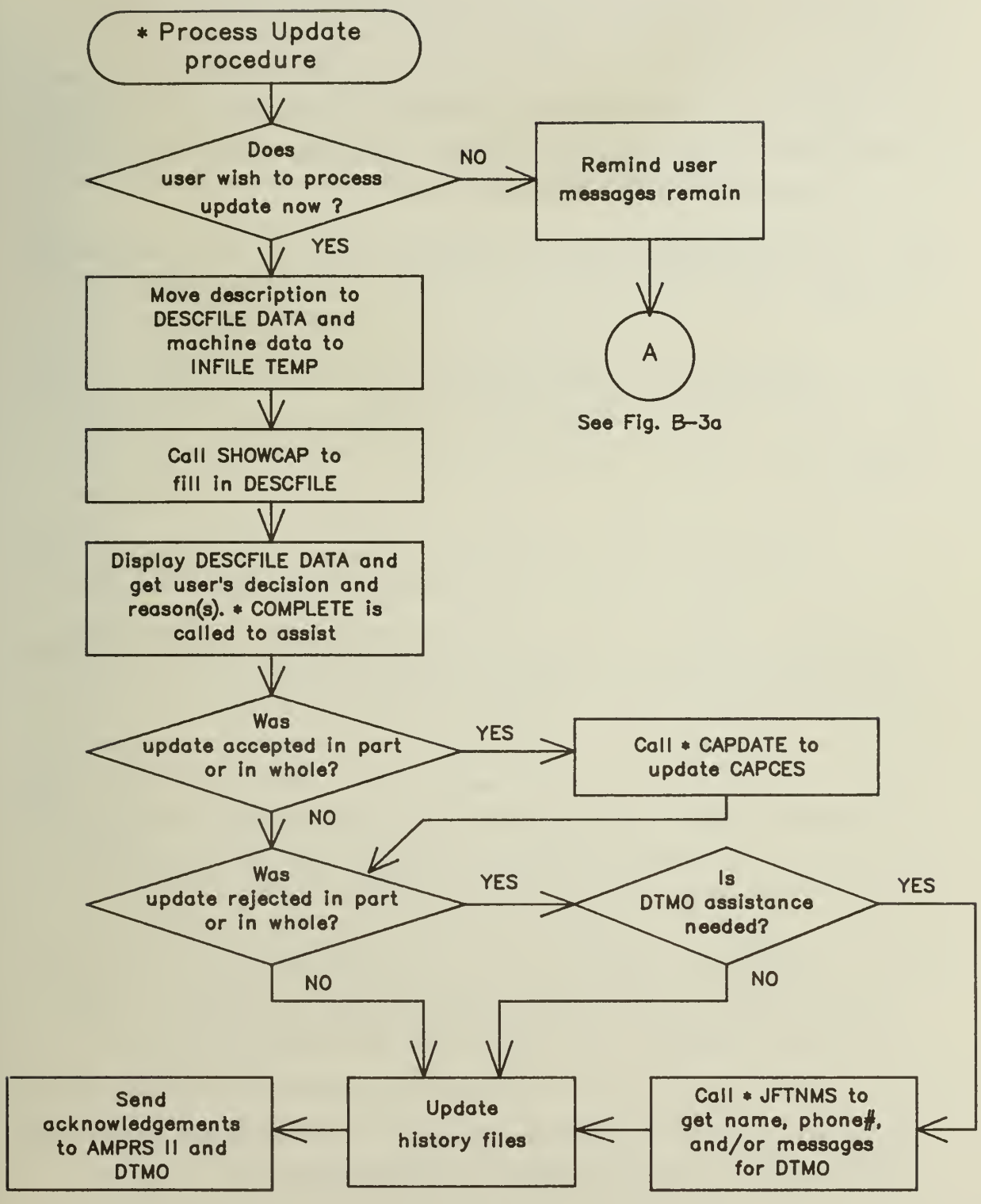

* Process Update procedure
Does user wish to process update now ?
NO
Remind user messages remain
YES
A
See Fig. B-3a
Move description to DESCFILE DATA and machine data to INFILE TEMP
Call SHOWCAP to fill in DESCFILE
Display DESCFILE DATA and get user's decision and reason(s). * COMPLETE is called to assist
Was update accepted in part or in whole?
YES
Call * CAPDATE to update CAPCES
NO
Was update rejected in part or in whole?
YES
Is DTMO assistance needed?
YES
NO
NO
Send acknowledgements to AMPRS II and DTMO
Update history files
Call * JFTNMS to get name, phone#, and/or messages for DTMO

Figure B-3b. Flow Chart for Processing Triggered Updates

transaction. The second item on this line is the number of items in this transaction. Based on this number, the number of lines in the record pertaining to this transaction is calculated. The DESCFILE pointer is moved ahead by this number of lines. Step 2 is carried out next. Otherwise, if the first item is the CAPCES user's user id, an unprocessed record is found. The number of items are moved into the top line of the file called NUMBERS DATA.

(4) If this is the first triggered update transaction of the RCV session, the machine displays

DTMS MESSAGES ARE WAITING.
DO YOU WANT MESSAGES DISPLAYED?

Otherwise, the prompt says:

WOULD YOU LIKE TO SEE THE NEXT UPDATE DESCRIPTION?

In either case, if the user answers negatively, A reminder that more updates remain is issued. The link to the DTMS space is detached and the RCV exits.

(5) Based on the known number of line in the record pertaining to this triggered update transaction and the current position of the DESCFILE line pointer, the bounds of the user description information and the machine input information for this update in DESCFILE STORE are determined. The former is copied into DESCFILE DATA and the latter is copied into INFILE TEMP.

(6) The information in DESCFILE DATA is not complete since some of the data needed to generate the description file reside on CAPCES. SHOWCAP is called to retrieve these items and put them in their proper places in DESCFILE DATA.

(7) The completed DESCFILE DATA is displayed to the user who can then choose to accept the entire transaction, defer it, reject it, or a combination of the three (a partial update) in which case, each item is treated individually.

(8) The program COMPLETE is called whose main purpose is to take DESCFILE DATA and embellish and use it both as acknowledgements and as entries in the history files. The output file of COMPLETE is called NEWDESC DATA. In

addition to information in DESCFILE DATA, NEWDESC DATA holds the date and time, the update number, and a message on how the user chose to treat the triggered update. In the case of a partial update, COMPLETE creates the partial input files depending on how each item was treated by the user (see step 12). In the case where any or all items were deferred or rejected, COMPLETE collects reasons for these actions and outputs them in NEWDESC DATA.

(9) The contents of NEWDESC DATA are copied into NEWDESC DTMS, which builds the acknowledgement message sent to DTMS. If the triggered update was not accepted in its entirety, step 11 is carried out next.

(10) If the triggered update has been accepted in whole, CAPDATE is called to update the CAPCES database using INFILE TEMP as input. NEWDESC DATA is copied into DESCFILE HISTORY (history file of partially or totally accepted triggered updates). Step 14 is carried out next.

(11) If a triggered update transaction was not totally accepted, NEWDESC DATA is copied into DESCFILE WAIT (history file of updates not totally accepted). If the transaction was entirely rejected, step 13 is carried out next. In the case of a partial update, step 12 is carried out. Otherwise (total deferral) INFILE TEMP is added into INFILE WAIT, since these data may be needed later. Step 14 is carried out next.

(12) This step is reached only in the case of a partial update. If any of the items were accepted, CAPDATE is called using INFILE PART as input (INFILE PART holds only the accepted updates.) NEWDESC DATA is added to DESCFILE HISTORY. If any items were deferred, their input records might be needed later. Therefore, INFILE DEFER is copied into INFILE WAIT. In order to insure that no items that were accepted or rejected are acted on later, INFILE records on these items are replaced by dummy records which only serve as place holders. If no items were rejected, the next step is 14.

(13) This step is reached if one or more items were rejected. Whether the user needs assistance of the DTMS management office is determined. If no assistance is needed, step 14 is carried out next. If assistance is needed, JFTNMS is called which prompts the user to leave a name and phone number or to describe what action is desired of DTMS. This information is added to NEWDESC DTMS.

(14) The acknowledgement messages are sent out. NEWDESC DATA is sent to the originating AMPRS II user, while NEWDESC DTMS is sent to DTMS. Finally, the first line of this transaction in DESCFILE STORE is flagged (see step 3) and the DESCFILE line pointer is advanced to the record pertaining the next triggered update transaction (or end of file). Step 2 is carried out next.

(15) Procedure -DONE is called when the DESCFILE STORE line pointer has reached the end of file. Its objective is to check if any new triggered updates remain to be picked up. Any new updates for CAPCES users will be stored in the DESCFILE file. The first line of this file contains a timestamp consisting of a date and time at which the DISCFILE file was updated last. This timestamp is compared with the one stored in TIME DATA. If they match, it indicates that no new updates have been made since the DESCFILE was examined the last time. "NO MORE MESSAGES" is displayed, the link to DTMS space is detached, and RCV ends. If the timestamps do not match, records on new updates have been added to the DESCFILE file. Triggered updates for which this user is responsible must be processed.

(16) DESCFILE file is then scanned for records on triggered updates containing the current CAPCES user id. If none are found, it means that the current CAPCES user is not responsible for any of the new triggered updates. "NO MORE MESSAGES" is displayed, the link to the DTMS space is detached, and RCV ends. Otherwise, step 17 is carried out next.

(17) As DESCFILE file is searched, records pertaining to transactions bearing the CAPCES user's user id are copied into a brand new DESCFILE STORE. That is, the old records have been deleted. When the DESCFILE file has been completely searched and DESCFILE STORE has been totally built, the DESCFILE line pointer is reset to the first line of the DESCFILE STORE.

(18) The new timestamp is copied into TIME DATA and the current CAPCES user id and this timestamp are sent to DTMS (placed in the ACK message file.) This serves as an acknowledgement of to DTMS management office and the update capture process stating that this portion of the DESCFILE file is already is already processed. (see B-1.) The next time new updates are captured and posted, the records with the current CAPCES user id can be purged from DESCFILE file.

Step 3 is then carried out.

### B-3. Description of the CAPCES Access Programs

There are two programs used in the AMPRS II-to-CAPCES link to access the CAPCES database. They are named CAPDATE and SHOWCAP.

#### a. CAPDATE Program.

The input data to this program is the INFILE. CAPDATE is a COBOL program which interfaces with FOCUS data PMMFILE. Maximum number of lines in an INFILE is five. Each line corresponds to updating one data item in PMMFILE. The format of each line is

KEYNR = _____, DATA_ITEM=DATA_VALUE,$

DATA_ITEM and DATA_VALUE may be different from each line. But KEYNR and its value is same throughout the INFILE.

List of DATA_ITEMs which is eligible to be updated by CAPDATE program is

```
SEGNAME=AMPERS1,PARENT=MAIN,SEGTYPE=U,$
        FIELD=DES_PERCENT,    ALIAS=DES_%,     USAGE=A3,$
        FIELD=CWE_AMT,        ALIAS=CWE,       USAGE=I8C,$
        FIELD=CONCOM_DT,      ALIAS=CONCEPT,   USAGE=I6YMTD,$
        FIELD=DES_COMP_DT,    ALIAS= ,         USAGE=I6YMTD,$
        FIELD=DES_ST_DT,      ALIAS=DES_SD,    USAGE=I6YMDT,$

SEGNAME=AMPERS2,PARENT=MAIN,SEGTYPE=U,$
        FIELD=HL_DES,         ALIAS=IH_DES,    USAGE=A1,$
        FIELD=MPC_DCD,        ALIAS= ,         USAGE=A1,$
        FIELD=CONS_COMP_DT,   ALIAS=CNCND,     USAGE=I6YMDT,$
        FIELD=BOD,            ALIAS=BEN_DCC_DT,USAGE=I6YMTD,$

SEGNAME=SUBPFILE,PARENT=MAIN,SEGTYPE=S1,$
        FIELD=SUBAMT,         ALIAS=,          USAGE=I8C,$
```

```
FIELD=SUBEXYR,          ALIAS= ,          USAGE=A4,$
FIELD=SUB_UNIT_MEA,     ALIAS=SUB_UM,     USAGE=A2,$
FIELD=SUBSCOPE,         ALIAS= ,          USAGE=I7C,$
FIELD=SUBITEM_DESC,     ALIAS=SDESC,      USAGE=A26,$
```

The hierarchical structure of CAPDATE program and function of each subroutine is as follows:

MAIN

It controls the whole program. Calls subroutines and checks possible errors.

INITIALIZATION

Opens input INFILE and read one line of INFILE. Then it moves item-name to name4 and move keynr to tlit11 and tlit21. Finally it opens the FOCUS PMMFILE.

UPDATING

Performs main body of update operation. This subroutine is called by MAIN and is repeatedly executed until end of file. It calls SHOW-COMD, GET-FIRST, BRANCHING, and READY-NEXT subroutines.

SHOW-COMD

Performs HLI command SHO.

GET-FIRST

Picks a PMMFILE MAIN segment whose keynr is same as keynr in the INFILE. This searching operation is done by HLI command NEX.

BRANCHING

Determines the PMMFILE segment in which the data-item of INFILE resides. There are three possible segments: AMPERS1, AMPERS2, or SUBPFILE. It also checks whether the data-item is in numeric mode or alphanumeric mode.

NUMFIELD

Because in-data is left justified and has a length of fifteen characters, it must be shortened and must be right justified. NUMFIELD corrects the length of the numeric in-data. It executes HLI command CHA which actually updates a numeric data in the PMMFILE.

ALPFIELD

Executes HLI command CHA which updates a alphanumeric data in the PMMFILE.

READY-NEXT

Closes and reopens the PMMFILE. Reads the next line of INFILE and moves item-name to name4.

CLOSE-COMD

Executes HLI command CLO. Closes INFILE.

### b. SHOWCAP Program

SHOWCAP is called to access CAPCES to retrieve information needed to complete the description file. The informations include PROJECT_YEAR, Type Funds name, PROJECT_DESCRIPTION, CATCD5, and original values of data-items in PMMFILE. The SHOWCAP program handles up to five data items at a time. The list of DATA_ITEMs which is eligible to be accessed by SHOWCAP program is the same as for CAPDATE. The hierarchical structure of SHOWCAP program and function of each subroutine is as follows:

MAIN

It controls the whole program. Calls subroutines and checks possible errors.

INITIALIZATION

Opens input-output file DESCFILE. Reads N, the second item in the first line which is the number of data items in the transaction. Reads keynr and moves it to tlit11. Read third line of DESCFILE. Fills up Type Funds name in fourth line. Opens the PMMFILE and calls SHOWING.

SHOWING

Picks up informations which resides in ZCPPFILE segment of PMMFILE. First it searches and picks up a MAIN segment whose keynr is equal to keynr from DESCFILE. Reads PROG_YEAR, PROJECT_DESCRIPTION and CATCD5 and fills up fifth and sixth lines of DESCFILE. Finally closes and reopens the PMMFILE.

FILLING

Performs main body of retrieval operation. This subroutine is called by MAIN and is repeatedly executed until end of file. It calls SHOW-COMD, SEARCHING, BRANCHING, and READY-NEXT subroutines.

SHOW-COMD

Performs HLI command SHO. First it subtracts one N and reads item-name from DESCFILE. Then performs SHO command.

SEARCHING

Picks a PMMFILE MAIN segment whose keynr is same as keynr in the INFILE. This searching operation is done by HLI command NEX.

BRANCHING

Determines the PMMFILE segment in which the DATA_ITEM of INFILE resides. There are three possible segments: AMPERS1, AMPERS2, or SUBPFILE. IT also checks whether the the DATA_ITEM is in numeric mode or alphanumeric mode.

NUMFIELD

It actually retrieves a numeric data in PMMFILE. This operation is done by using HLI command NEX.

ALPFIELD

Executes HLI command NEX which retrieves a alphanumeric data in the PMMFILE.

READY-NEXT

Closes and reopens the PMMFILE. Reads the next line of INFILE and moves item-name to name4.

CLOSE-COMD

Executes HLI command CLO. Closes INFILE.

| BIBLIOGRAPHIC DATA SHEET | 1. Report No. UIUCDCS-R-83-1142 | 2. | 3. Recipient's Accession No. |
|---|---|---|---|

4. Title and Subtitle

MUTUAL CONSISTENCY MAINTENANCE IN A PROTOTYPE DATA TRAFFIC MANAGEMENT SYSTEM

5. Report Date

6.

7. Author(s) G.G. Belford, J.W.S. Liu, D. Cho, P. Cotten, S. England, J.D. Goldstein, S.C. Hwung, K.A. Kaufman, C.K. Kim, J. Leo, A.P. Manolas, A. Moon, A.D. Smet, Y.L. Yan, G.L. Robinson and R.L. Lapp

8. Performing Organization Rept. No.

9. Performing Organization Name and Address

Department of Computer Science, University of Illinois
1304 W. Springfield Ave, Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

Department of the Army, Construction Engineering Res. Lab.
Interstate Research Park, Newmark Drive, P.O. Box 4005
Champaign, Illinois 61820

13. Type of Report & Period Covered

14.

15. Supplementary Notes

16. Abstracts

This report describes the mutual consistency maintenance mechanism used in a prototype Data Traffic Management System (DTMS) which was designed and implemented to integrate several independent and stand-alone data systems owned by the U.S. Army Corp of Engineers (CE). The major design constraints were that no modifications to the data systems involved were allowed and that an update to each duplicate copy was to be handled as automatically as possible and in a manner approved by the administrator of the data system containing the copy. The purpose of this work was to develop and test operating procedures which can be expanded and modified so as to be applicable in the implementation of an operational version of DTMS. The capabilities of the prototype DTMS and the Mutual Consistency Maintenance Subsystem (MCMS) are discussed. The linkage directory used to support mutual consistency maintenance and the mutual consistency maintenance mechanism are described.

17. Key Words and Document Analysis. 17a. Descriptors

Mutual Consistency Maintenance
Distributed Data Bases

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

| 18. Availability Statement | 19. Security Class (This Report) UNCLASSIFIED | 21. No. of Pages 52 |
|---|---|---|
| | 20. Security Class (This Page UNCLASSIFIED | 22. Price |

UNIVERSITY OF ILLINOIS-URBANA
3 0112 103707102